1991

On Alternate Solid Representations and Their Uses

Christoph M. Hoffmann
Purdue University, cmh@cs.purdue.edu

George Vaněček

Report Number:
91-019

ON ALTERNATE SOLID REPRESENTATIONS AND THEIR USES

Christoph M. Hoffmann
George Vanecek, Jr.

CSD-TR-91-019
March 1991

On Alternate Solid Representations and Their Uses*

Christoph M. Hoffmann† George Vaněček, Jr.‡

Computer Science Department
Purdue University
West Lafayette, IN 47907

Abstract

We discuss a number of surface and solid representations especially suited to specific tasks arising in product design and analysis. The Brep-index is a data structure well-suited to support collision detection and analysis in mechanical simulation. The dimensionality paradigm provides a simple and effective mechanism for representing constrained surfaces precisely. The skeleton, finally, is an informationally-complete solid representation that shows promise for automated mesh generation, feature recognition and extraction, and for geometric tolerancing.

1 Introduction

The two major representation schemata in geometric and solid modeling are the volume-based constructive solid geometry schema (CSG) and the surface-based boundary-representation schema (Brep). Both have specific strengths and weaknesses, yet are sufficiently universal that entire solid modeling systems have been built based on one or the other schema. In our view, there are certain applications in manufacturing and electronic prototyping that put exacting demands on the underlying solid representation that warrant exploring alternate data structures for representing solids. In this paper, we discuss several such alternate solid representations.

*Presented at the DARPA Manufacturing Meeting, University of Utah, 1991.
†Supported in part by ONR Contract N00014-90-J-1599 and by NSF Grant CCR 86-19817.
‡Supported by NSF Grant CCR 86-19817.

An emerging application in manufacturing and electronic prototyping is the simulation of dynamical systems based on geometric data. Such simulations are used in two distinct ways.

1. In the analysis of mechanisms with fixed kinematic behavior, the geometric data is used only initially to establish mass properties and to formulate holonomic constraints on the dynamical system. Such simulations can safely abstract each moving body as an oriented mass point with suitable inertial properties. Therefore, these simulations make no exceptional demands on the underlying geometric representation.

2. When the kinematics of the dynamical system is not known a-priori, for example when simulating the motion of objects in the presence of obstacles, or when simulating bodies in intermittent contact at unpredictable loci, then the dynamics simulation must inquire at each time step whether two bodies are about to collide, and if so, perform a geometric analysis of the local geometry in the vicinity of the contact points. The results of the geometric analysis are then used to formulate a system of differential equations for evaluating the dynamical consequence of the collision. In such applications, the massive number of collision queries and analyses suggests using specialized solid representations that facilitate and optimize the geometric computations.

In Section 2 we present a data structure, the *Brep index*, that is especially well suited to the task of geometric collision testing and analysis. In particular, the data structure integrates volume-based and boundary based representations in a manner that does not sacrifice efficiency or robustness. The technical details of constructing the Brep index are very similar to the conversion from Brep to CSG.

The Brep index is constructed for polyhedral solids. In principle, the Brep index exists also for solids with curved surfaces, as long as halfspace queries can be supported that permit testing whether a point is outside, inside, or on the boundary of the halfspace defined by the curved surface. In particular, implicit surfaces given by an equation of the form $f(x, y, z) = 0$ support such queries. However, algorithms for constructing the Brep index for curved solids are not yet known.

A number of solid queries and operations are highly desirable, yet there are at present no algorithms that exhibit compelling efficiency or competence. Such operations include finite-element meshing of solids, feature extraction and recognition, and geometric tolerancing. It is plausible that CSG and Brep representations complicate potential algorithms for these problems. In contrast, the *skeleton*, discussed in Section 4, is an informationally-complete solid representation that appears to facilitate all those operations. While there is presently

little hard evidence for this expectation, a number of authors have independently articulated this view, as summarized later.

Constructing an exact representation of the skeleton involves finding *equi-distance* surfaces. An equi-distance surface is the locus of all points in 3-space that have equal distance from two given geometric objects. For example, the equi-distance surface of a plane and a point not in the plane is a paraboloid of revolution. Except for very simple geometric objects, equi-distance surfaces cannot be represented exactly using traditional techniques. In fact, the difficulty of deriving exact representations for equi-distance surfaces and other geometrically constrained surfaces has seriously impeded constructing and using the skeleton.

We have discovered an alternate curve and surface representation that permits constructing equi-distance surfaces and other geometrically constrained surfaces with great ease. Moreover, the representation can be interrogated efficiently and uniformly. This approach to curve and surface representation is called the *dimensionality paradigm*, because the representation defines a manifold in n-space that is projected into a subspace of dimension 2 or 3. We discuss the dimensionality paradigm in Section 3. Note that the approach delivers the basic tools that enable us to construct skeleton representations of curved solids in 3-space.

2 The Brep Index

Many problems in geometric modeling and its applications have to determine which vertex, edge, or face of a solid contains a given point or intersects a given line [41, 40]. This determination is called the *point/solid* and *line/solid classification* problems. The classifications can be performed directly on boundary-based representations (Breps), but the operations are then slow and complex to implement. To understand why, note that a Brep provides ample information about the surface points, and the topological adjacencies between vertices, edges and faces, but that the Brep does not give a spatial order to the vertices, edges and faces that would allow quick classification. Preprocessing the vertices, edges and faces of the Brep into a spatially ordered data structure can therefore greatly speed up and simplify the point/solid and line/solid classification problems. The Brep index is such a data structure [42].

In Subsection 2.1, we outline an application from manufacturing that motivated the creation of the Brep index; i.e., collision detection and analysis of moving objects. In Subsection 2.3 we describe the Brep index in detail.

2.1 Collision Detection of Moving Objects

Consider a computer simulation of a system of rigid bodies in Newtonian physics [22, 23]. The dynamic motion of all bodies is described by a system of ordinary differential equations, and the integration of these equations determines the position, orientation, velocity, etc., of each body over time. We can then record this information at discrete time steps.

At each such moment in time, we can check whether any two bodies of the simulated physical system collide, and whether any two bodies that were in contact remain in contact. If no collisions occur and no contacts disappear, then the system of differential equations is integrated normally. However, when unforeseen events occur, then the time of the event as well as its characteristics must be determined, and resolved. The unforeseen events are collisions, and they are detected by a geometric computation that determines whether two objects interpenetrate at the current time, and if so, how. Note that the loss of contact is primarily a physical event, and that it is detected by monitoring contact forces.

Assume that we have detected an interpenetration of two bodies. We can then back up the system to the previous time step, and based on the geometric characteristics of the penetrating bodies, the depth of penetration, and the relative velocity, we can estimate a smaller time step which brings us to the moment when the bodies first come into contact. Having so isolated in time the moment of collision, possibly iterating the backup and estimation procedure, we can then analyze the geometric characteristics of the collision and its consequences for the state of the physical system, as well as the required alterations of the system of differential equations.

Detecting interpenetration and collision, as well as the subsequent analysis of the nature of the penetration or contact, is a geometric computation that can be reduced to a number of point/solid and line/solid classifications [44]. In the Newton/ProtoSolid simulation system, this computation is implemented as a query dialogue between the dynamics simulation system in charge of formulating and integrating the differential equations of motion, and the geometric modeling system in charge of all geometric aspects of the simulation process. Figure 1 shows the interface between the geometric modeling system and the dynamic simulation system that enables this query dialogue. Note that the two systems are autonomous, running on separate machines, and that only a minimum of information is exchanged in this dialogue. The simulation system keeps track of the position and orientation, the velocity and the angular velocity of each object, as well as the constraints and contact forces. The shape of each object is maintained only by the geometric system. At each time step, the simulation system gives the geometric system the names, positions, and orientations of two

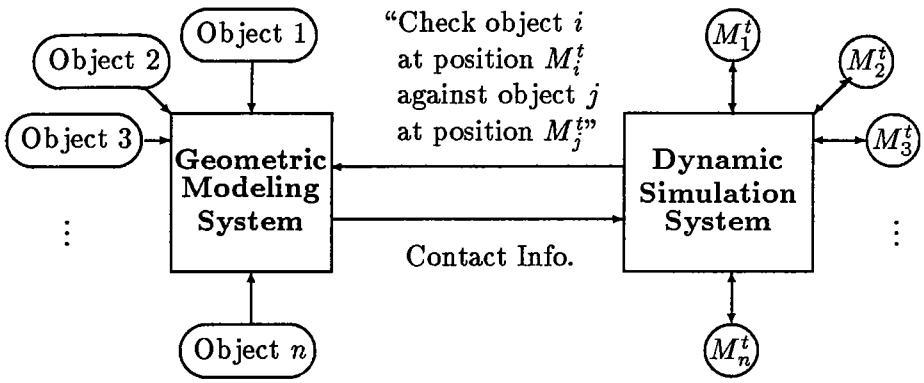

Figure 1: The interface to the geometric modeling system for performing static collision detection and analysis.

objects and requests a collision test.

2.2 Spatial Analysis of Two Objects

From the above discussion, we can formulate the geometric collision testing problem as follows. We are given the Breps of two objects, each described in a local coordinate frame whose origin is the centroid of the object. We also know the position and orientation of each body by a 4x4 transformation matrix. The matrix specifies the mapping of the local frame of reference of the object, to a global, inertial frame of reference (GFR). With each object so mapped to the GFR, we need to check whether the objects penetrate, touch or are spatially disjoint. When two objects penetrate, we further need to estimate the time of collision. When two objects touch, we need to find all contact points and return a finite set of well-chosen contact points, each described by the five tuple:

$$(p_1, x_1, p_2, x_2, \mathbf{n}_2),$$

where p_1 and p_2 are the contact points, each in the object's local frame of reference; where x_1 and x_2 are the topological entities that contain the points, i.e., the respective vertex, edge or face; and where $\mathbf{n}_2$ is the contact normal, given in the local frame of reference of object 2. See also Figure 2.

In addition to collisions, objects can be in contact over an extended period of time. Such temporary contact occurs when two objects are pushing against each other or are sliding across each other. In this case, the relative velocity of the contacting bodies in the direction of the contact normal is zero. For

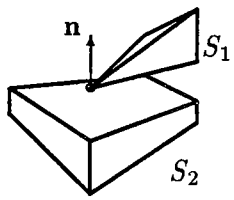

Figure 2: Example of a vertex/face collision.

the geometric modeling system to correctly analyze the contacts, it must be informed by the dynamics simulation system as to which of the contacts are collisions and which ones are temporary contacts. Note that two objects can be simultaneously in temporary contact at one point and colliding at another point.

We need to distinguish the type of contact because the contact normal computation may differ in the two cases. Assume object 1 slides with vertex v across face f of object 2. It is possible that at some moment v is on an edge of f. In that case, we require by temporal coherence that the contact normal is the normal of f. If, on the other hand, v collides with an edge of f, then the normal on which the collision computation is based should account for the adjacent face, and the normal will bisect the exterior angle between the two faces. Furthermore, the relative position of objects in prolonged contact has a tendency to drift slightly over time, due to numerical noise in the integration. Therefore, we allow temporary contacts to interpenetrate somewhat without reporting a collision, and this contact tolerance could well differ from the collision tolerance.

We analyze the spatial relationship of two objects by classifying all their vertices and edges. For polyhedral objects, this is both necessary and sufficient for detecing all forms of contact and interpenetration. The cost of classifying the vertices and edges of object 1 against object 2 is

$$O(v_1 V(B_2) + e_1 E(B_2)),$$

where v_1 and e_1 is the number of vertices and edges of object 1, to be classified, and $V(B_2)$, and $E(B_2)$ is the average cost of classifying a vertex or an edge against the Brep B_2 of object 2. Our goal is to use a classification scheme that yields the lowest cost functions V and E.

2.3 The Structure of the Brep index

The Brep index is a ternary-tree data structure that represents a recursive, mutli-dimensional partitioning of space. Associated with an internal tree node n

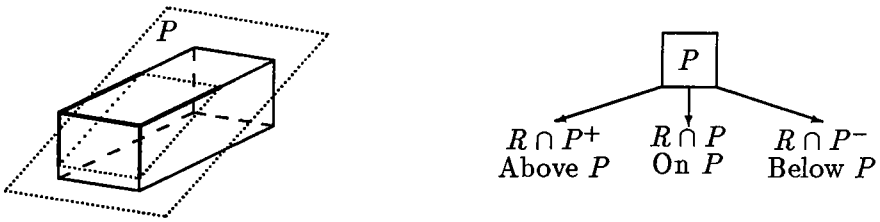

Figure 3: Region R cut by a plane P is partitioned into three subregions that compose the three subtrees of node represeting R.

is a cut plane P that partitions the region represented by n into three subregions (refer to Figure 3). The three subregions are represented by the three children of n, and are labelled according to whether the subregion lies above, on or below P. The resulting spatial partition is multi-dimensional because the leaves of the tree represent zero, one, two and three-dimensional regions. If the region represented by a node with cut plane P has dimension d, then the subregions above and below P also have dimension d, but the subregion on P has dimension $d-1$. The Brep index generalizes the binary space partition (BSP) tree [17, 28].

As an example of a Brep index, consider the boundary representation of a tetrahedral solid shown in Figure 4. The figure shows the Brep index, and the Brep with labelled vertices, edges and faces. A cut plane P_i in the Brep index is a support plane of face F_i, and has a normal vector that points away from the solid. Since the tetrahedron is a convex solid, there is only a single region that lies inside the solid, and this region is represented by the solidly black node in the figure.

If a solid is convex, then we can always construct a Brep index of size

$$v + e + f,$$

where v, e, and f are the number of vertices, edges and faces, respectively. For example, the tetrahedon of Figure 4 has four vertices, six edges and four faces, and a tree with 14 internal nodes. For nonconvex solids, the size grows as a result of the global fragmentation by an additional $O(G)$ that is due to the fragmentation of the solid's surface by the cut planes. A Brep index for a solid is not unique, but depends on the ordering of the cut planes. Changing the order gives an equivalent index, possibly of a different size. In the best case, the additional cost $O(G)$ is the difference between $v + e + f$ and the number of leaves in smallest such index tree. We are currently studying compression techniques by which any Brep index can be converted to one of minimum size.

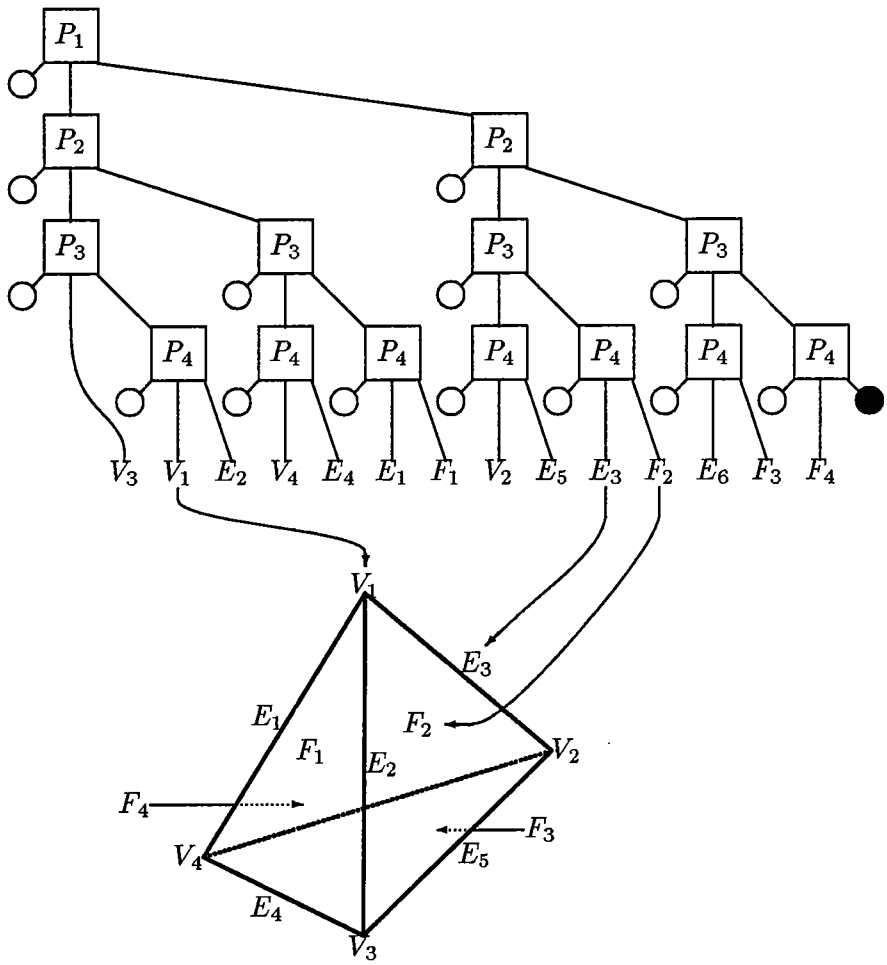

Figure 4: Example Brep index for a tetrahedral solid. The white circles indicate outside regions; the black circle indicates the single inside region.

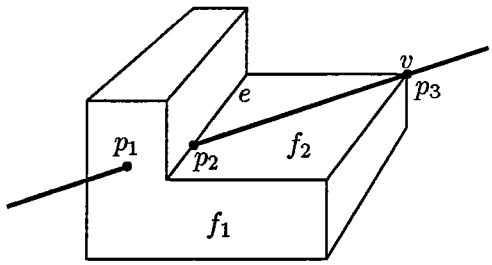

Figure 5: An example of a line/solid classification.

2.4 Point and Line Classification

We explain how to solve the point/solid and the line/solid classification problems using the Brep index.

To classify a point, start at the root of the tree and traverse a path from the root to a leaf. The path ends at a leaf representing a vertex, edge, or a face of the Brep, or the labels **inside** or **outside**. At each internal node n, the point is checked against the cut plane to determine whether the point is above, on, or below the plane. Accordingly, the search continues with the left, middle, or right child of n. Thus, the cost of classifying a point is proportional to the length of the path.

A line segment is classified as follows. Beginning at the root, the segment is checked against the cut plane. If the segment intersects, it is partitioned with the part above being forwarded to the left descendant, the intersection point to the middle descendant, and the part below the cut plane being sent to the right descendant. In this way, the segment is filtered through the tree and partitioned. After each part of the induced segment partition is classified, the information is passed back up to the root and recombined. Figure 5 shows a line segment classified against a solid resulting in the classification sequence:

$$[\mathbf{out}, (f_1, p_1), \mathbf{in}, (e, p_2), f_2, (v, p_3), \mathbf{out}].$$

Consistent and correct classification is a necessity. Consider the classification of the line segment ℓ shown in Figure 6. Cut planes P_1 and P_2 intersect each other at vertex v of the solid. The line segment crosses the plane P_1 at the point q with a very small angle. Even though the line segment can be arbitrarily close to v, the point q may be arbitrarily far away from v. Therefore, when the line segment ℓ is split by P_1 at q, the segments below, on and above P_1 are all found to lie **outside** the solid, even though the line passes the vertex within close tolerance.

When the distance from v to ℓ is less than the specified tolerence, the line segment classification should determine that v lies on ℓ and that the projection

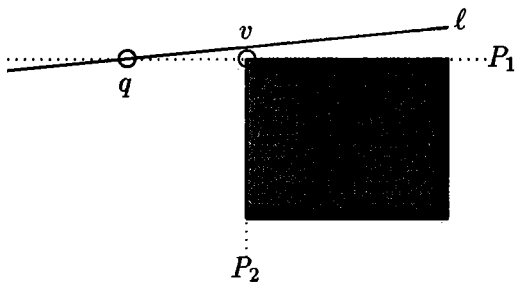

Figure 6: Splitting ℓ by P_1 misses the detection of v as lying within tolerence of ℓ.

of v onto ℓ rather than q should partition the segment. To achieve this, we change the way in which the line is split at internal nodes. We define a zone around the cut plane by considering the offsets of the plane in either direction, at distance equal to some tolerance. By intersecting the line with the two offsets, the segment is partitioned using an interval rather than an intersection point. Figure 7 shows the offsets of planes P_1 and P_2. The interval obtained from the two offsets of P_1 now intersects the region around v. This indicates that v should be projected to ℓ, thereby obtaining the point r closest to v on ℓ.

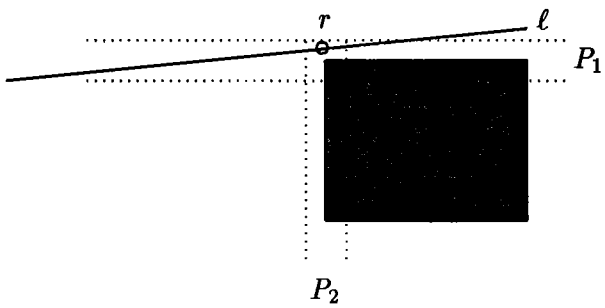

Figure 7: Finding the closest point, r, on ℓ to v using offset planes.

3 The Dimensionality Paradigm

The dimensionality paradigm provides a method for representing surfaces defined through constraints. It simplifies representing mathematically complicated surfaces, and it permits generic algorithms for manipulating them. The method is of interest because it is an enabling technology for constructing promising solid representations such as the skeleton discussed later in Section 4.

3.1 Background and Motivation

There are many conceptual geometric operations on curves and surfaces that are easily explained in intuitive terms and are simple to understand. For instance, given two surfaces f and g, consider all points in space that have equal minimum distance from the given surfaces. Such points form the *equidistance* or *Voronoi surface* of f and g. Despite this conceptual simplicity, it is by no means a trivial undertaking to represent surfaces so derived in exact mathematical terms, in a manner suitable for computation.

Another example is the *rolling-ball blend*, a blending surface between given surfaces f and g obtained as follows: Roll a sphere such that it maintains contact with both f and g, at all times. Consider the volume in space traversed by the sphere. Then the rolling ball blend is defined by the surface of that volume.

The basic property of these and similar other surface definitions is that the new surface to be defined can be expressed in terms of one or more base surfaces and a number of intuitive geometric constraints. The dimensionality paradigm provides a means for expressing such a definition by a system of nonlinear equations. The ease of definition requires defining the surface in a space of higher dimension. More precisely, the required surface is the natural projection of a 2-manifold in n-space, where $n > 3$. The extra dimensions may be point coordinates on the base surface(s), distances, or other quantities used to express the constraints that must be obeyed.

When the base surfaces are algebraic, and this is normally the case in geometric and solid modeling, then the resulting system of equations can be processed by a number of symbolic computation algorithms that eliminate all additional variables and arrive at a single implicit equation for the surface. Such an approach is not normally tractable, for it routinely arrives at elimination problems that are well beyond what hardware and software can deliver in the foreseeable future. It is not uncommon to exceed hundreds of mega bytes of memory in the course of the symbolic computation that should eventually produce such a closed-form representation. Worse yet, a number of elimination algorithms require exact arithmetic. Therefore, we work directly with the system of equa-

tions, bypassing expensive elimination calculations altogether.

We have developed a number of algorithms for surfaces defined using the dimensionality paradigm. The algorithms are general and do not require knowledge of the geometric nature of the surfaces. Therefore, they will work unchanged on offsets, on blends, and on equidistance surfaces. Some algorithms are local in nature. That is, given a point on the surface, the algorithms will explore the surface in the vicinity of the point. Such algorithms can and have been globalized, by suitably embedding the exploration into a spatial grid that coordinates the computation.

3.2 An Example Definition

We consider the definition of an equidistance surface as an example of the dimensionality paradigm. We are given two implicit surfaces $f(x, y, z) = 0$ and $g(x, y, z) = 0$. Using a declarative style, we can describe the equidistance surface as follows:

1. Let $p = (x, y, z)$ be a point on the equidistance surface. Moreover, let $p_f = (u_1, v_1, w_1)$ be a point at minimum distance from p on f, and let $p_g = (u_2, v_2, w_2)$ be a point at minimum distance from p on the surface g. Then:
2. The point p_f satisfies the equation of f, and the point p_g satisfies the equation of g.
3. The distance (p, p_f) is d and is equal to the distance (p, p_g).
4. The line $\overline{p, p_f}$ is normal to f at p_f.
5. The line $\overline{p, p_g}$ is normal to g at p_g.

Note that assertion 1. declares the names of nine variables, the coordinates of three points, whereas assertions 2.–5. simply state the geometric relationships that these points must satisfy.

In order to obtain the equational representation of the equidistance surface, we translate the assertions 2.–5., using the variable names of 1. We obtain in sequence:

$$f(u_1, v_1, w_1) = 0 \quad (1)$$

$$g(u_2, v_2, w_2) = 0 \quad (2)$$

$$(x - u_1)^2 + (y - v_1)^2 + (z - w_1)^2 = d^2 \quad (3)$$

$$(x - u_2)^2 + (y - v_2)^2 + (z - w_2)^2 = d^2 \quad (4)$$

$$
\begin{aligned}
[x-u_1, y-v_1, z-w_1] \cdot [-f_{v_1}, f_{u_1}, 0] &= 0 && (5)\\
[x-u_1, y-v_1, z-w_1] \cdot [f_{w_1}, 0, -f_{u_1}] &= 0 && (6)\\
[x-u_1, y-v_1, z-w_1] \cdot [0, -f_{w_1}, f_{v_1}] &= 0 && (7)\\
[x-u_2, y-v_2, z-w_2] \cdot [-g_{v_2}, g_{u_2}, 0] &= 0 && (8)\\
[x-u_2, y-v_2, z-w_2] \cdot [g_{w_2}, 0, -g_{u_2}] &= 0 && (9)\\
[x-u_2, y-v_2, z-w_2] \cdot [0, -g_{w_2}, g_{v_2}] &= 0 && (10)
\end{aligned}
$$

Subscripting, as in f_{u_1}, denotes partial differentiation.

Equations (1)–(3) are quite clear. Equations (5)–(7) together express assertion 4., since the three vectors

$$
\begin{aligned}
&[-f_{v_1}, f_{u_1}, 0]\\
&[f_{w_1}, 0, -f_{u_1}]\\
&[0, -f_{w_1}, f_{v_1}]
\end{aligned}
$$

are tangent to f and span the tangent space as long as p_f is not a singular point on the surface. Similarly, equations (8)–(10) together express assertion 5.

Note that if f is given in parametric form, then equations (1) and (5)–(7) have to be adjusted. This is routine and shows that the methodology is independent of the representation of the base surfaces.

The entire system of equations defines a manifold in 10-dimensional space.[1] The projection of that manifold into the (x, y, z)-subspace is the equidistance surface. In principle, an implicit equation could be derived by eliminating the variables $\{u_1, v_1, ..., w_2, d\}$ from the system, but in almost all cases this computation is not tractable.

3.3 Faithful Systems

The equation systems formulated by the dimensionality paradigm entail certain additional point sets that are unwanted because they do not reflect the geometric intent. The origin of these additional solutions is due to possible interdependence of the individual equations at certain points in space. For example, if p_f is a singular point, then equations (5)–(7) vanish. In consequence, the system also defines a manifold that projects to the equidistance surface of g and the singular point.

Extraneous solutions present in the system can be excluded by introducing certain additional equations, [24]. These new equations use one or more additional variables that no longer have a geometric meaning, but are used to express

[1] note that d is a variable

inequalities through equations. This "trick" is common in automated geometry theorem proving. Note that all extraneous solutions can be so eliminated. The equations added reflect a generic method, but must account for the geometry of the original system.

3.4 Available Infrastructure

We have developed a considerable body of algorithmic infrastructure to deal with surfaces defined by systems of nonlinear equations. The following algorithms are now available:

1. Given two surfaces and an initial point, evaluate their intersection; see [4; 18, Chapter 6; 20]. The algorithm is robust and can evaluate very high-degree surface intersections without significant precision problems.

2. Given a surface and an initial point, evaluate locally the curvatures, [12], and give a local parametric or local explicit surface approximant of arbitrary contact order, [11, 19].

3. Given a surface and an initial point, globally approximate the surface; [11].

These algorithms are not confined to algebraic equation systems, and work well as long as the nonlinear equations of the system are continuously differentiable. They are very efficient.

Less efficient are the known techniques for finding initial points. When nothing is known about the system, then generic techniques such as [1, 2, 6, 7, 27] can be applied. Usually, however, the geometric intent of the system is known and leads to good initial estimates for starting points that can be refined using Newton iteration. Moreover, some of the search techniques from CAGD are often applicable. That is, by raising the dimension of the ambient space by 1, all equations can be rewritten in Bernstein-Bèzier form after which domain reduction is applied using the convex hull property. To implement this, one has to exercise care not to obtain an exponential growth in the number of control points.

4 The Skeleton as Solid Representation

The *interior skeleton* of a three-dimensional solid is the locus of the centers of all inscribed maximal spheres. A sphere is *maximal* if there is no other

sphere that contains it completely. Other names for the skeleton include *medial-axis transform* and *symmetric-axis transform.* While little is known about the skeleton in three dimensions, there has been work on the two-dimensional version of the problem, and it has been argued that the skeleton in 2D is useful for generating finite-element meshes [3, 29, 39].

We have what we believe is the first complete algorithm to construct the skeleton of three-dimensional CSG domains, [21], that are constructed from the standard primitives with the usual regularized Boolean operations. Other algorithms for 3D domains have been attempted, but are approximate only because of the difficult surfaces that occur in the skeleton; [5]. Our algorithm became possible through our work on the dimensionality paradigm.

If we know for each point p of the skeleton the radius of the maximal inscribed sphere centered at p, then the skeleton becomes an informationally-complete solid representation. This suggests to view the skeleton as a four-dimensional object. Each skeleton point has three spatial coordinates and one coordinate giving the (minimum) distance to the boundary. We have begun an investigation of the question which operations on solids are facilitated by such a skeletal representation. The many applications of the 2D skeleton suggest that this representation of 3D domains could have a number of important applications, including preparing domains for mesh generation, recognizing features, and performing offsetting operations. Since this work is still in the beginning stages, we comment only briefly on these topics below.

In the following, we assume that we are given a skeleton in four dimensions, composed of faces, edges and vertices. The spatial point coordinates are named x, y, z, and the distance coordinate is named r. Thus, the point (x, y, z, r) of the skeleton corresponds to the inscribed sphere of the 3D domain that is centered at (x, y, z) and has radius r. In general, this sphere contributes two points on the domain boundary in 3D.

For simplicity we illustrate our ideas with pictures of skeletons in three dimensions, of 2D domains, consisting of edges and vertices. In this case, the point coordinates are (x, y, r), where r is the distance coordinate. Figure 8 shows the domain and the skeleton in the traditional, two-dimensional way, whereas Figure 9 illustrates the three-dimensional view of the skeleton. Given a skeleton, we ask which operations on solids simplify with this representation schema.

4.1 Offset Operations on Solids

Given a solid as its skeleton, the interior offset of the solid is obtained by translating the skeleton, as a rigid object, in the $-r$ direction by the offset distance.

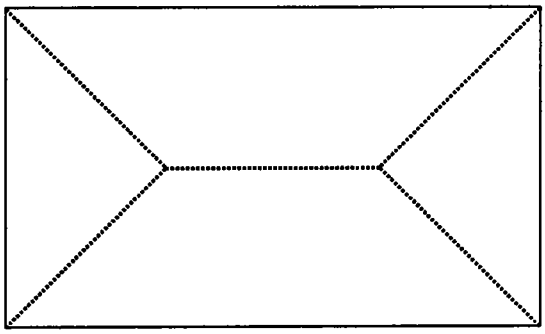

Figure 8: A 2D domain and its Interior Skeleton

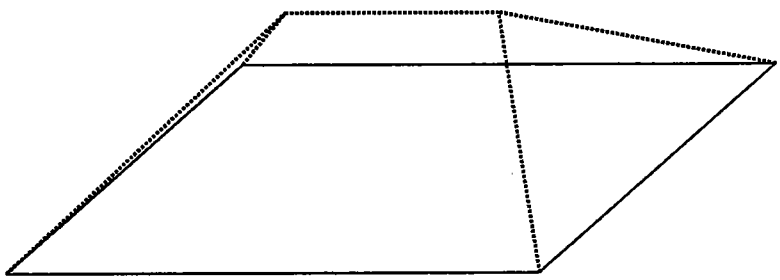

Figure 9: The 2D Domain and its Interior Skeleton Viewed in 3D

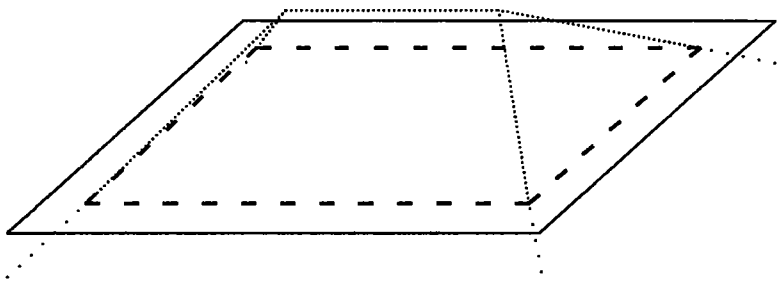

Figure 10: Interior Offsetting on the Skeleton

Thereafter, those points of the translated skeleton that have a negative distance coordinate are clipped. Thus, if the offset distance is d, the skeleton S is changed to the skeleton S', where

$$S' = \{(x, y, z, r - d) \,|\, (x, y, z, r) \in S \text{ and } r \geq d\}$$

Figure 10 illustrates the idea for 2D domains: The skeleton of the domain shown in Figure 9 has been "pushed down" by the offset distance. Now the skeleton penetrates the plane of the domain at the four corner points of the interior domain offset (dashed), and the skeleton parts that lie below the plane are indicated by a different dot density. Those parts would be clipped.

The exterior offset to a solid is obtained similarly by "lifting" the skeleton, by the offset distance, in the positive r direction. This is not quite right, because we are missing the counterpart of the clipping procedure that was part of the interior offsetting operation. Merely lifting the skeleton results in self-intersections of the offset boundary. The situation can be remedied in one of two ways:

1. If we consider both the interior and the exterior skeleton, then raising the interior skeleton implies lowering the exterior skeleton. Hence, clipping the exterior skeleton tells us where to modify the lifted interior skeleton.

2. We can construct a geometric object that represents *all* offsets, including their topology. The skeleton is then a section of this "holographic" representation by a halfspace $r \geq$ const.

Both ideas need further exploration. In fact, the second approach extends classical work in descriptive geometry of the 19th century that studied noneuclidean distance functions. Note that in the example of Figure 9 the exterior domain offset is indeed obtained by lifting the skeleton, because the 2D domain is convex.

4.2 Geometric Tolerancing

In the simplest approach to geometric tolerancing one imagines the boundary having a certain "thickness." The skeleton is promising in this regard because of its ability to represent the offset by translation. More complicated tolerancing involves spatial interrelationships between solid features. Not much is known about the behaviour of the skeleton in this regard. However, the merging step of divide-and-conquer algorithms for constructing Voronoi diagrams is related in a technical sense, and has been explored in 2D for polygonal domains.

4.3 Feature Recognition

Form feature recognition and extraction is difficult in boundary representations because features consist of a number of faces that are not always adjacent and are spatially interrelated. CSG representations would do better if they could be converted to a canonical form that is explicitly related to the feature. This latter approach has not had broad success. For one, it is observed that the design process that results in the CSG tree may have proceeded without regard to specific features of interest. Also, it is sometimes argued that the concept of feature is only relative to a specific view of the product. For example, a feature meaningful to the manufacturing process need not be functionally significant, and vice versa. Thus, it is held that it is not possible to define *ab initio* all features of a particular design, and this then motivates searching for an algorithmic way to recognize and extract features.

The skeleton representation appears promising for feature recognition because the shape of each skeleton face depends on the spatial relationship of separate, possibly distant, boundary elements. Thus, a feature often depends on a single skeleton face. For example, a boss can be characterized by a single face of the interior skeleton. A slot would be characterized by a face of the exterior skeleton. See also the contribution by Prinz et al. in these proceedings.

4.4 Mesh Generation

Systematic finite-element mesh generation has been based on three fundamental approaches:

1. Superimpose a spatial subdivision, meshing interior cells in a standard way, and do special processing near the boundary; e.g., [38, 41].

2. Beginning from an initial point sampling, on the domain boundary, construct a Delaunay triangulation that respects the boundary of the domain; e.g., [9].

3. Partition the domain by its skeleton and some additional edges or faces into simple subdomains, and mesh the subdomains in a manner that is compatible across the interfaces between them; e.g., [3, 29, 39].

It is argued in [39] that Approach (1) does not address the essence of the geometric problem and merely postpones it to those spatial cells that intersect the boundary. While Approach (2) does well in 2D and can guarantee favorable aspect ratios, no similar guarantees exist in the 3D case, as argued in [3]. Using the Approach (3), some impressive results have been achieved in the 2D case, including the ability to recognize completely automatically areas of constriction in which the mesh by necessity would have to be finer than elsewhere. Approach (2) achieves this only with help of a user-defined density function. Skeleton based approaches to the 3D problem thus appear very promising,

5 Summary

We have discussed the Brep index, a data structure for supporting interference testing and analysis in mechanical simulations. If we consider curved solids whose faces are trimmed implicits, then a Brep index also exists. Studying the algorithm for constructing the Brep-index of polyhedral solids [42] reveals that the algorithm for the Brep index is almost identical to algorithms for converting a Brep to CSG. In [36, 37] such an algorithm is attempted, for solids using the standard CSG primitives. The algorithm is not complete, but points out that a Brep index for CSG objects would require auxiliary cut surfaces that are not supporting any face of the Brep. Figure 11 shows a two-dimensional example. Here, the domain boundary includes a circular arc. In CSG, it can be constructed from the union of a rectangle and a disk, but one corner of the rectangle must be removed by a third primitive that leaves no trace on the domain boundary. The cut line X could bound this "invisible" primitive.

We have also discussed the dimensionality paradigm, a methodology for representing complex curves and surfaces by systems of nonlinear equations. For this shape representation there exists an extensive and competent algorithmic infrastructure that has been briefly outlined. The dimensionality paradigm is a major tool that enables us to construct the skeleton, an informationally-complete solid representation. The skeleton promises to facilitate a number of operations on solids such as mesh generation, geometric tolerancing, offsetting, and feature recognition.

Although Brep and CSG have been the dominant solid representations for many years, it is our view that alternate solid representations offer significant advantages in specific important applications. The importance of these appli-

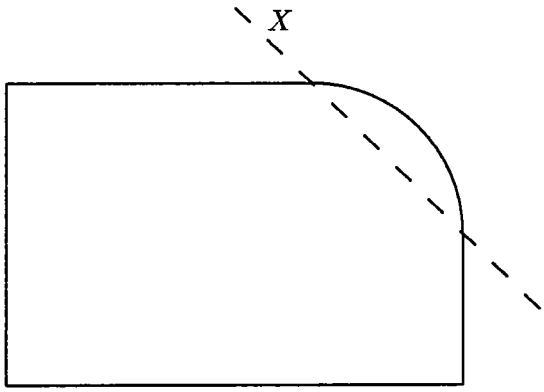

Figure 11: A Curved Object Requiring An Auxiliary Cut Line X

cations warrant carefully exploring the potential of such alternate solid representations and using them in practice.

6 References

1. E. Allgower, K. Georg, R. Miranda (1990), "Computing Real Solutions of Polynomial Systems," Preprint, Math. Dept., Colorado State Univ., Ft. Collins.

2. E. Allgower and S. Gnutzmann (1987), "An Algorithm for Piecewise Linear Approximation of Implicitly Defined Two-Dimensional Surfaces," *SIAM J. Numer. Anal. 24*, 452–469.

3. C. Armstrong, T. Tam, D. Robinson, R. McKeag, M. Price (1990), "Automatic Generation of Finite Element Meshes," SERC ACME Directorate Research Conference, England.

4. C. Bajaj, C. Hoffmann, J. Hopcroft, R. Lynch (1988), "Tracing Surface Intersections," *CAGD 5*, 285–307.

5. A. Bowyer (1991), "On Skeleton Construction," IMPA Workshop on Geometric Modeling, Rio de Janeiro.

6. B. Buchberger (1985), "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory," in *Multidimensional Systems Theory*, N. K. Bose, ed., D. Reidel Publishing Co., 184–232.

7. B. Buchberger, G. Collins, and B. Kutzler (1988), "Algebraic Methods for Geometric Reasoning," *Annl. Reviews in Comp. Science*, Vol. 3, 85–120.

8. A. Cayley (1848), "On the Theory of Elimination," *Cambridge and Dublin Math. J. 3*, 116–120.

9. P. Chew (1989), "Guaranteed-Quality Triangular Meshes," Tech. Rept 89-893, Comp. Sci., Cornell University.

10. E.-W. Chionh (1990), "Base Points, Resultants, and the Implicit Representation of Rational Surfaces," PhD Diss., Comp. Sci., Univ. Waterloo, Canada.

11. J.-H. Chuang (1990), "Surface Approximations in Geometric Modeling," PhD Dissertation, Comp. Sci., Purdue University.

12. J.-H. Chuang and C. Hoffmann (1990), "Curvature Computations on Surfaces in n-Space", Report CER-90-34, Comp. Sci., Purdue Univ.

13. V. Chandru, D. Dutta, C. Hoffmann (1990), "Variable Radius Blending with Cyclides," in *Geometric Modeling for Product Engineering* K. Preiss, J. Turner, M. Wozny, eds., North Holland; 39–57.

14. S. Coquillart (1987), "Computing offsets of B-spline curves," *CAD 19*, 305–309.

15. R. Farouki (1986), "The Approximation of Nondegenerate Offset Surfaces," *Comp. Aided Geometric Design 3*, 15–43.

16. R. Farouki and C. Neff (1989), "Some Analytic and Algebraic Properties of Plane Offset Curves," Rept. RC 14364, IBM Yorktown Heights.

17. H. Fuchs, Z. Kedem, and B. Naylor (1980), "On visible surface generation by a priori tree structures," *Proc. SIGGRAPH '80*, 124–133.

18. C. Hoffmann (1989) *Geometric and Solid Modeling*, Morgan Kaufmann Publishers, San Mateo, Cal.

19. C. Hoffmann (1989) "Algebraic and Numerical Techniques for Offsets and Blends," in *Computations of Curves and Surfaces*, W. Dahmen, M. Gasca, S. Miccelli, and L. Schumaker, eds., Kluwer Academic Publishers; 499–528.

20. C. Hoffmann (1990), "A dimensionality paradigm for surface interrogation," *CAGD 7*, 517–532.

21. C. Hoffmann (1990), "How to Construct the Skeleton of CSG Objects," *4th IMA Conf. Math. of Surfaces*, Univ. of Bath, England.

22. C. Hoffmann and J. Hopcroft (1987), "Simulation of physical systems from geometric models", *IEEE J Robotics and Automation RA-3*, 194–206.

23. C. Hoffmann and J. Hopcroft (1988), "Model generation and modification for dynamic systems from geometric data," in *CAD Based Programming for Sensory Robots*, B. Ravani, ed, NATO ASI Series F50, Springer Verlag, 481–492.

24. C. Hoffmann and P. Vermeer (1991), "Eliminating Extraneous Solutions in Curve and Surface Operations," *Intl. J. Comp. Geom. Applic.*, to appear.

25. J. Hoschek (1985), "Offset curves in the plane," *CAD 17*, 77–82.

26. J. Hoschek (1988), "Spline approximation of offset curves," *CAGD 5*, 33–40.

27. A. Morgan (1987), *Solving Polynomial Systems Using Continuation for Scientific and Engineering Problems*, Prentice-Hall, Englewood Cliffs, N.J.

28. B. Naylor (1990), "Binary Space Partitioning Trees as an Alternative Representation of Polytopes," *Computer-Aided Design 22*, 250–252.

29. N. Patrikalakis and H. Gürsoy (1990), "Shape Interrogation by Medial Axis Transform," Memo 90-2, Ocean Engr. Design Lab., MIT.

30. J. Pegna (1988), "Variable Sweep Geometric Modeling," PhD Diss., Mech. Engr., Stanford Univ.

31. B. Pham (1988), "Offset approximation of uniform B-splines," *CAD 20*, 471–474.

32. J. Rossignac and A. Requicha (1984), "Constant Radius Blending in Solid Modeling," *Comp. Mech. Engr. 3*, 65–73.

33. J. Rossignac, A. Requicha (1984), "Offsetting operations in solid modeling," *CAGD 3*, 129–148.

34. S.E.O. Saeed, A. de Pennington, J.R. Dodsworth (1988), "Offsetting in geometric modeling," *CAD 20*, 67–74.

35. T. Sederberg (1983), "Implicit and Parametric Curves and Surfaces for Computer Aided Geometric Design," *Ph.D. Diss.*, Mech. Engr., Purdue University

36. V. Shapiro and D. Vossler (1990), "Brep to CSG Conversion I: Construction and Optimization of CSG Representations," Rept. CPA89-3a, Mech. and Aerosp. Engr., Cornell University.

37. V. Shapiro and D. Vossler (1990), "Brep to CSG Conversion II: Efficient Representations of Planar Solids," Rept. CPA89-4a, Mech. and Aerosp. Engr., Cornell University.

38. M. Shepard, F. Guerinoni, J. Flaherty, R. Ludwig, P. Baehmann (1988), "Finite Octree Mesh Generation for Automated Adaptive three-dimensional Flow Analysis," *Proc. 2nd Intl. Conf Num. Grid Generation in Comp. Fluid Mech.*, Miami, 709–718.

39. V. Srinivasan, L. Nackman, J. Tang, S. Meshkat (1990), "Automatic Mesh Generation using the Symmetric Axis Transformation of Polygonal Domains," Rept. RC 16132, IBM Yorktown Heights.

40. R. Tilove (1981), "Line/Polygon Classification: A Study of the Complexity of Geometric Computation," *IEEE Comp. Graphics Appl. 1*, 75–88.

41. G. Vaněček, Jr. (1990), "Towards Automatic Grid Generation using Binary Space Partition Trees," Rept. CER-90-6, Comp. Sci, Purdue University.

42. G. Vaněček, Jr. (1991), "Brep-Index: A Multi-dimensional Space Partitioning Tree," *1st ACM/SIGGRAPH Symp. Solid Modeling Found. and CAD/CAD Appl.*, Austin, Texas.

43. G. Vaněček, Jr. (1989), "ProtoSolid: An inside look," Rept. CER–89–26, Comp. Sci, Purdue University.

44. G. Vaněček, Jr. (1991), "A Data Structure for Analyzing Collisions of Moving Objects," *IEEE 24th Annl. Hawaii Intl. Conf. Syst. Sci.*, Vol I, 671–680.